IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appellants::	Gerard Chauvel, et al.	Confirmation No.:	1444
		Group Art Unit:	2185
Serial No.:	10/631,185	Examiner:	Aran P. Savla
Filed:	July 31, 2003	Atty. Docket No.:	TI-35431 1962-05410
For:	Dirty Cache Line Write Back Policy Based On Stack Size Trend Information		

APPEAL BRIEF

Mail Stop Appeal Brief—Patents
Commissioner for Patents
PO Box 1450
Alexandria, VA 22313-1450

September 4, 2009

Sir or Madam:

Appellants hereby submit this Appeal Brief in connection with the above-identified application. A Notice of Appeal was electronically filed on August 14, 2009.

TABLE OF CONTENTS

I. REAL PARTY IN INTEREST

The real party in interest is Texas Instruments Incorporated, a Delaware corporation, having its principal place of business in Dallas, Texas.

II. RELATED APPEALS AND INTERFERENCES

Appellants are unaware of any related appeals or interferences.

III. STATUS OF THE CLAIMS

Originally filed claims:	1-20
Claim cancellations:	None
Added claims:	None
Presently pending claims:	1-20
Presently appealed claims:	1-20

IV. STATUS OF THE AMENDMENTS

No claims were amended after the Final Office action dated March 16, 2009.

V. SUMMARY OF THE CLAIMED SUBJECT MATTER

This section provides a concise explanation of the subject matter defined in each of the independent claims and separately argued dependent claims involved in the appeal, referring to the specification by page and line number or to the drawings by reference characters as required by 37 C.F.R. § 41.37(c)(1)(v). Each element of the claims is identified with a corresponding reference to the specification or drawings where applicable. Note that the citation to passages in the specification or drawings for each claim element does not imply that the limitations from the specification and drawings should be read into the corresponding claim element. Also note that these specific references are not exclusive; there may be additional support for the subject matter elsewhere in the specification and drawings.

The current application discloses using stack trend information to determine when to write to main memory and when to restrict writing to main memory from the cache. Preferably, when the stack is increasing, dirty cache lines are written to main memory, but when the stack is decreasing, dirty cache lines are not written to main memory.

Claim 1 recites a method of managing memory Fig.3; Fig.5. The method includes determining stack trend information using current and future stack operating instructions p.12 ℓ.9–10, 22–23. The method further includes reducing data traffic between various levels of a memory based on the trend information p.13 ℓ.6–p.14 ℓ.13.

Claim 7 recites a method of managing memory Fig.3; Fig.5. The method includes determining stack trend information using current and future stack operating instructions p.12 ℓ.9–10, 22–23. The method further includes utilizing the trend information to reduce data traffic between various levels of a memory p.13 ℓ.6–p.14 ℓ.13. The levels of memory includes a cache memory containing multiple cache lines and a main memory p.7 ℓ.18–20; Fig.1 refs. 14, 16. The trend information is used to restrict writing dirty cache lines from cache memory to main memory when the trend information indicates the stack is decreasing p.11 ℓ.15–22; p.13 ℓ.23–p.14 ℓ.13.

Claim 9 recites a method of managing memory Fig.3; Fig.5. The method includes determining stack trend information using current and future stack operating instructions

p.12 ℓ.9–10, 22–23. The method further includes determining if the size of the stack is going to increase as a result of future instructions p.12 ℓ.23–p.13 ℓ.5. The method further includes utilizing the trend information to reduce data traffic between various levels of a memory p.13 ℓ.6–p.14 ℓ.13. The method further includes determining if a line is written back including analyzing the trend information and examining a dirty cache line to determine which word of the dirty cache line is going to be written to p.13 ℓ.16–19; p.13 ℓ.6–p.14 ℓ.13.

Claim 11 recites a computer system Fig.1 ref.10. The system includes a processor Fig.1 ref.12; p.6 ℓ.6; p.5 ℓ.11–p.6 ℓ.5, and a memory coupled to the processor Fig.1 refs.14, 16; p.6 ℓ.6–7. The system further includes a stack that exists in memory and contains stack data Fig.1 ref.32; p.6 ℓ.13–p.7 ℓ.1. The system further includes a memory controller coupled to the memory Fig.1 ref.26; Fig.2 ref.26; p.7 ℓ.22–p.9 ℓ.18, and trend logic Fig.4 ref.21; p.12 ℓ.1–p.13 ℓ.5. The processor executes instructions Fig.1 ref.12; p.5 ℓ.14, and the trend logic provides trend information about the stack to the controller p.12 ℓ.10–13; Fig.4 refs.21, 26. The trend information about the stack is based on at least one future instruction p.12 ℓ.9–10, 13–15, 18–23.

Claim 15 recites a computer system Fig.1 ref.10. The system includes a processor Fig.1 ref.12; p.6 ℓ.6; p.5 ℓ.11–p.6 ℓ.5, and a memory coupled to the processor Fig.1 refs.14, 16; p.6 ℓ.6–7. The system further includes a stack that exists in memory and contains stack data Fig.1 ref.32; p.6 ℓ.13–p.7 ℓ.1. The system further includes a memory controller coupled to the memory Fig.1 ref.26; Fig.2 ref.26; p.7 ℓ.22–p.9 ℓ.18, and trend logic Fig.4 ref.21; p.12 ℓ.1–p.13 ℓ.5. The processor executes instructions Fig.1 ref.12; p.5 ℓ.14. The trend logic provides trend information about the stack to the controller p.12 ℓ.10–13; Fig.4 refs.21, 26. The trend information about the stack is based on at least one future instruction p.12 ℓ.9–10, 13–15, 18–23. The memory includes a cache memory containing multiple cache lines and a main memory p.7 ℓ.18–20; Fig.1 refs. 14, 16. The trend information is used to restrict writing dirty cache lines from cache memory to main memory when the trend information indicates the stack is decreasing p.11 ℓ.15–22; p.13 ℓ.23–p.14 ℓ.13.

Claim 16 recites a computer system, Fig.1 ref.10. The system includes a processor Fig.1 ref.12; p.6 ℓ.6; p.5 ℓ.11–p.6 ℓ.5, and a memory coupled to the processor Fig.1 refs.14, 16; p.6 ℓ.6–7. The system further includes a stack that exists in memory and contains stack data Fig.1 ref.32; p.6 ℓ.13–p.7 ℓ.1. The system further includes a memory controller coupled to the memory Fig.1 ref.26; Fig.2 ref.26; p.7 ℓ.22–p.9 ℓ.18, and trend logic Fig.4 ref.21; p.12 ℓ.1–p.13 ℓ.5. The processor executes instructions Fig.1 ref.12; p.5 ℓ.14, and the trend logic provides trend information about the stack to the controller p.12 ℓ.10–13; Fig.4 refs.21, 26. The trend information about the stack is based on at least one future instruction p.12 ℓ.9–10, 13–15, 18–23, and the memory includes a cache memory and a main memory, p.7 ℓ.18–20; Fig.1 refs. 14, 16. The cache memory contains a dirty cache line p.6 ℓ.3–5, and the dirty cache line is written to main memory if the trend information indicates the stack is increasing p.13 ℓ.21–23; p.9 ℓ.4–10.

Claim 17 recites a method including issuing a write request to a cache memory Fig.4 ref.74. The cache memory includes multiple cache lines p.7 ℓ.18–20; Fig.1 refs. 14, 16. The method further includes determining whether the write request refers to a predetermined word within a dirty cache line Fig.4 refs.76, 78. The method further includes determining whether to write the dirty cache line to main memory based on whether the size of a stack is increasing or decreasing Fig.4 refs.80, 82, 84.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Whether claims 1-4, 6-10 and 15-16 are obvious due to Shen in view of Flake with PC Guide offered as extrinsic evidence.

Whether claims 12 and 14 are obvious due to Shen in view of O'Connor (U.S. Patent No. 6,026,485 "O'Connor").

Whether claim 5 is obvious due to Shen in view of Flake in further in view of Ebrahim (U.S. Patent No. 5,893,121 "Ebrahim").

Whether claims 11 and 13 are anticipated by Shen (U.S. Patent No. 5,687,336 "Shen") with PC Guide ("The PC Guide: The Memory Controller") used as external evidence.

Whether claims 17-20 are anticipated by Flake (U.S. Patent No. 7,065,613 "Flake").

VII. ARGUMENT

The claims do not stand or fall together. Instead, Appellants present separate arguments for various independent and dependent claims. After a brief discussion of the cited art, each of these arguments is separately argued below and presented with separate headings as required by 37 C.F.R. § 41.37(c)(1)(vii).

A. Discussion of the Cited Art

1. Shen (U.S. 5,687,336)

At col.4 ℓ.48–54, Shen discusses updating a stack pointer,

> Three-port adder 40 is used to generate the address of the top of the stack (TOS) directly without calculating the new stack pointer. The old stack pointer 12 from register file 10, along with the increment amount from increment logic 20 and the segment base address 24 are added together without having to generate the new stack pointer.

Shen continues at col.4 ℓ.54–56, "The new stack pointer SP' is not generated until the end of the pipeline, when the new stack pointer is written to the register file 10."

2. Flake (U.S. 7,065,613)

At col.6 ℓ.39–44 Flake discloses skipping writes of old data lines because a tag value points to an unused section of the cache:

> This old tag value corresponds to addresses 9C–9F, which are in the unused section of the cache. The algorithm skips the write of the old data line, sets the line 1 tag to the new value, and clears the dirty flag. The new tag value corresponds to the unused section of the stack so the read of the line from main memory is skipped. The processor write then proceeds and the dirty flag is set.

Indeed, such behavior is the novelty of Flake, as discussed in the abstract, "More specifically, in one embodiment, the invention prevents writing old line data to main memory when the old line data represents a currently unused area of the cache." The intention of implementing such behavior is discussed at col.6 ℓ.56–63:

> The cache operation allows for single and multiple-object deletion by changing the stack pointer, for quick multiple object deletion. No special push or pop indications are required and, if used, such indications would

> prevent optimization for multiple-object deletion. Further, multiple-object creation through a stack pointer is inherently supported with the described cache.

At col.6 ℓ.13, 32, Flake names the combined actions of writing to an address and then setting the stack pointer a "stack-growth write."

B. Obviousness Rejections of Claims 1–4, 6–10, and 15–16 Due to Shen in View of Flake with PC Guide Offered as Extrinsic Evidence

1. Claims 1–4 and 6–10

Independent claim 1 recites, in part, "reducing data traffic between various levels of a memory based on the [stack] trend information." Independent claims 7 and 9 recite a similar limitation: "utilizing the [stack] trend information to reduce data traffic between various levels of a memory." However, Examiner erred in rejecting the claims because the cited references fail to teach or suggest the limitation. On page 5 of the Final Office Action, Examiner notes that Shen does not disclose this information, and Examiner cites Flake at col.6 ℓ.32–44 as allegedly teaching the limitation. At the cited location, Flake states "This old tag value corresponds to addresses 9C–9F, *which are in the unused section of the cache*." At col.5 ℓ.38–40 Flake explains: "By virtue of knowing when an access was or will be to *the unused portion of a stack*, the algorithm reduces writes and reads to main memory." However, determining access to an unused portion of a stack fails to teach or suggest determining the trend information of the stack.

Examiner states on page 5 of the Final Office Action, "It should be noted that the decision to skip reading the line from main memory is based on the stack operation." Appellants respectfully disagree. First, determining unused sections of cache is not a stack operation. Second Appellants have not claimed a generic "stack operation;" rather, Appellants have claimed "stack trend information." Furthermore, no other art of record teaches or suggests the quoted limitations as required by the claims. For at least this reason, Examiner erred in rejecting independent claims 1, 7, and 9, along with dependent claims 2–4, 6, 8, and 10, and the claims are allowable over Shen in view of Flake with PC Guide offered as extrinsic evidence.

2. Claim 15

Independent claim 15 recites, in part, "wherein the trend information is used to restrict writing dirty cache lines from cache memory to main memory when the trend information indicates the stack is decreasing." However, Examiner erred in rejecting the claims because the cited references fail to teach or suggest the limitation. On page 10 of the Final Office Action, Examiner notes that Shen does not disclose this information, and on page 11 of the Final Office Action, Examiner cites Flake at col.6 ℓ.29–31 as allegedly teaching the limitation. At the cited location, Flake states "The next processor operation is a deletion of a group of stack objects achieved by changing the stack pointer to A8. No cache functions occur." However, multiple object deletion by changing the stack pointer fails to teach or suggest restrict writing dirty cache lines from cache memory to main memory when the trend information indicates the stack is decreasing. Indeed at col.6 ℓ.56–62, Flake states, "The cache operation allows for single and multiple-object deletion by changing the stack pointer, for quick multiple object deletion. No special push or pop indications are required and, if used, such indications would prevent optimization for multiple-object deletion." Furthermore, no other art of record teaches or suggests the quoted limitations as required by the claim. For at least this reason, Examiner erred in rejecting independent claim 15, and the claim is allowable over Shen in view of Flake with PC Guide offered as extrinsic evidence.

3. Claim 16

Independent claim 16 recites, in part, "wherein the dirty cache line is written to main memory if the trend information indicates the stack is increasing." However, Examiner erred in rejecting the claim because the cited references fail to teach or suggest the limitation. On page 12 of the Final Office Action, Examiner cites Flake at col.6 ℓ.13–28 as allegedly teaching the limitation. At the cited location, Flake states, "The cache logic determines that this write is a miss by virtue of its tag value, binary 10011, and index value, binary 1, not matching line 1. The miss begins the line replacement algorithm of FIG. 5.." However, teaching writing a dirty cache line based on a cache miss does not teach or suggest writing a dirty cache line based on an increasing stack. Flake characterizing a cache miss as a "stack-growth write" does not create a causal

relationship between an increasing stack and writing a dirty cache line. Furthermore, no other art of record teaches or suggests the quoted limitations as required by the claim. For at least this reason, Examiner erred in rejecting independent claim 16, and the claim is allowable over Shen in view of Flake with PC Guide offered as extrinsic evidence.

C. Obviousness Rejections of Claims 12 and 14 due to Shen in view of O'Connor

Claims 12 and 14 depend from independent claim 11. Examiner cites the same art against the similar portions of the claims. Furthermore, O'Connor fails to teach the deficiencies of Shen. Therefore, the reasoning below applies, and claims 12 and 14 are allowable over Shen in view of O'Connor.

D. Obviousness Rejection of Claim 5 due to Shen in view of Flake in further in view of Ebrahim

Claim 5 depends from independent claim 1. Examiner cites the same art against the similar portions of the claims. Furthermore, Ebrahim fails to teach the deficiencies of Shen. Therefore, the reasoning above applies, and claim 5 is allowable over Shen in view of Flake in further in view of Ebrahim.

E. Anticipation Rejections of Claims 11 and 13 over Shen

Independent claim 11 recites, in part, "wherein the trend information about the stack is based on at least one future instruction." However, Examiner erred in rejecting the claims because the cited references fail to teach or suggest the limitation. On page 3 of the Final Office Action, Examiner cites Shen at col.3 ℓ.65–col.4 ℓ.7; col.4 ℓ11–15, 36–60; Figs.2, 4 as allegedly teaching the limitation. At the cited location, col.4 ℓ.36–50, Shen teaches the new stack pointer created from an increment value, the old stack pointer, and a segment base address. However, the new stack pointer is "not generated until the end of the pipeline." As such, the new stack pointer cannot be based on at least one future instruction because at the end of the pipeline no future instructions will exist. Thus, Shen fails to teach or suggest the quoted limitation as required by the claim. Furthermore, no other art of record teaches or suggests the quoted limitations as required by the claims. For at least this reason, Examiner erred in rejecting independent claim 11, along with dependent claim 13, and the claims are allowable over Shen.

F. Anticipation Rejections of Claims 17–20 over Flake

Independent claim 17 recites, in part, "determining whether to write the dirty cache line to main memory based on whether the size of a stack is increasing or decreasing." However, Examiner erred in rejecting the claims because the cited references fail to teach or suggest the limitation. On page 4 of the Final Office Action, Examiner cites Flake at col.6 ℓ.13–28 as allegedly teaching the limitation. At the cited location, Flake states, "The cache logic determines that this write is a miss by virtue of its tag value, binary 10011, and index value, binary 1, not matching line 1. The miss begins the line replacement algorithm of FIG. 5.." However, teaching writing a dirty cache line based on a cache miss does not teach or suggest writing a dirty cache line based on an increasing or decreasing stack. Flake characterizing a cache miss as a "stack-growth write" does not create a causal relationship between an increasing stack and writing a dirty cache line. Furthermore, no other art of record teaches or suggests the quoted limitations as required by the claims. For at least this reason, Examiner erred in rejecting independent claim 17, along with dependent claims 18–20, and the claims are allowable over Flake.

VIII. CONCLUSION

For the reasons stated above, Appellants respectfully submit that the rejections should be reversed. In the course of the foregoing discussions, Appellants may have at times referred to claim limitations in shorthand fashion, or may have focused on a particular claim element. This discussion should not be interpreted to mean that the other limitations can be ignored or dismissed, or that limitations from the specification can be imported into the claims. The claims must be viewed as a whole, and each limitation of the claims must be considered when determining the patentability of the claims. Moreover, it should be understood that there may be other distinctions between the claims and the prior art which have yet to be raised, but which may be raised in the future.

It is believed that no extensions of time or fees are required, beyond those that may otherwise be provided for in attachments accompanying this document. However, in the event that additional extensions of time are necessary to allow consideration of this document, such extensions are hereby petitioned under 37 C.F.R. § 1.136(a), and any

fees required (including fees for net addition of claims) are hereby authorized to be charged to Texas Instruments Incorporated's Deposit Account No. 20-0668.

Respectfully submitted,

/Tim D. Chheda/

Tim D. Chheda
PTO Reg. No. 60,752
CONLEY ROSE, P.C.
(713) 238-8000 (Phone)
(713) 238-8008 (Fax)
ATTORNEY FOR APPELLANTS

IX. CLAIMS APPENDIX

1. A method of managing memory, comprising:

determining stack trend information using current and future stack operating instructions; and

reducing data traffic between various levels of a memory based on the trend information.

2. The method of claim 1, wherein determining the trend information includes examining future instructions to determine if the size of the stack is going to decrease as a result of future instructions.

3. The method of claim 2, wherein a predetermined number of instructions are used in determining stack trend information.

4. The method of claim 3, wherein the number of predetermined instructions is at least two.

5. The method of claim 3, wherein the cache memory maintains a single dirty cache line for stack data.

6. The method of claim 3, wherein if a dirty cache line needs to be written back, then analyzing the trend information, which includes determining which word of the dirty cache line is going to be written to.

7. A method of managing memory, comprising:

determining stack trend information using current and future stack operating instructions; and

utilizing the trend information to reduce data traffic between various levels of a memory, wherein the levels of memory comprise a cache memory containing multiple cache lines and a main memory, and wherein the trend information is used to restrict writing dirty cache lines from cache memory to main memory when the trend information indicates the stack is decreasing.

8. The method of claim 1, wherein determining the trend information includes examining future instructions to determine if the size of the stack is going to increase as a result of future instructions.

9. A method of managing memory, comprising:

determining stack trend information using current and future stack operating instructions, including determining if the size of the stack is going to increase as a result of future instructions;

utilizing the trend information to reduce data traffic between various levels of a memory; and

determining if a line is written back including analyzing the trend information and including examining a dirty cache line to determine which word of the dirty cache line is going to be written to.

10. The method of claim 9, wherein the dirty cache line is written from a cache memory to a main memory.

11. A computer system, comprising:

a processor;

a memory coupled to the processor;

a stack that exists in memory and contains stack data;

a memory controller coupled to the memory;

trend logic;

wherein the processor executes instructions;

wherein the trend logic provides trend information about the stack to the controller; and

wherein the trend information about the stack is based on at least one future instruction.

12. The computer system of claim 11, further comprising an instruction decoder comprising a first portion that decodes current instructions and a second portion that decodes future instructions.

13. The computer system of claim 12, wherein the trend logic determines a net stack trend based on current instruction and future instruction information coming from the decoder.

14. The computer system of claim 12, wherein the second portion of the decoder is adjusted so that the number of future instructions that are decoded equals at least two.

15. A computer system, comprising:

a processor;

a memory coupled to the processor;

a stack that exists in memory and contains stack data;

a memory controller coupled to the memory;

trend logic;

wherein the processor executes instructions;

wherein the trend logic provides trend information about the stack to the controller;

wherein the trend information about the stack is based on at least one future instruction; and

wherein the memory includes a cache memory containing multiple cache lines and a main memory, and wherein the trend information is used to restrict writing dirty cache lines from cache memory to main memory when the trend information indicates the stack is decreasing.

16. A computer system, comprising:

a processor;

a memory coupled to the processor;

a stack that exists in memory and contains stack data;

a memory controller coupled to the memory;

trend logic;

wherein the processor executes instructions;

wherein the trend logic provides trend information about the stack to the controller;

wherein the trend information about the stack is based on at least one future instruction; and.

wherein the memory includes a cache memory and a main memory, and wherein the cache memory contains a dirty cache line, and wherein the dirty cache line is written to main memory if the trend information indicates the stack is increasing.

17. A method, comprising:

issuing a write request to a cache memory, wherein the cache memory includes multiple cache lines;

determining whether the write request refers to a predetermined word within a dirty cache line; and

determining whether to write the dirty cache line to main memory based on whether the size of a stack is increasing or decreasing.

18. The method of claim 17, further comprising determining whether the write request will be to the end of a dirty cache line.

19. The method of claim 18, wherein the stack size is increasing and the dirty cache line is written to a main memory.

20. The method of claim 18, wherein the stack size is decreasing and the dirty cache line is retained in the cache memory.

X. EVIDENCE APPENDIX

Studying for the A+, Network+ or Security+ exams? Get over **2,600 pages** of FREE study guides at CertiGuides.com!

Join the PC homebuilding revolution! Read the all-new, FREE 200-page online guide: How to Build Your Own PC!

NOTE: Using robot software to mass-download the site **degrades the server and is prohibited.** See here for more.

Find The PC Guide helpful? Please consider a donation to The PC Guide Tip Jar. Visa/MC/Paypal accepted.

Take a virtual vacation any time at DesktopScenes.com - beautiful low-cost screensavers of my nature photography!

NEW! All of my art photos can now be viewed online for FREE in either the Flash gallery or HTML gallery!

[The PC Guide | Systems and Components Reference Guide | System Memory | Memory Speed, Access and Timing]

The Memory Controller

Every computer has within it a hardware logic circuit called the *memory controller*. This is the part of the system that, well, controls the memory. It generates the necessary signals to control the reading and writing of information from and to the memory, and interfaces the memory with the other major parts of the system. The memory controller is normally integrated into the system chipset.

Next: Memory Access and Access Time

Home - Search - Topics - Up

XI. RELATED PROCEEDINGS APPENDIX

None.